

DM6816 Driver for Windows 98/NT4/2000/XP

User's Manual

Version 4.1



SWM-640010008
Rev. B



RTD Embedded Technologies, INC.

103 Innovation Blvd.
State College, PA 16803-0906

Phone: +1-814-234-8087

FAX: +1-814-234-5218

E-mail

sales@rtd.com

techsupport@rtd.com

web site

<http://www.rtd.com>

Revision History

03/26/2003	Version 4.0 released 2000/XP support added. Interface of function OpenBoard6816 changed
08/06/2003	revision A issued Documented for ISO9000
09/24/2003	Revision B issued Version 4.1 released Company name changed Windows 95 support removed All function and files names *6816 changed to *6816

DM6816 Driver for Windows 98/NT4/2000/XP

Published by:

RTD Embedded Technologies, Inc.

103 Innovation Blvd.

State College, PA 16803-0906

Copyright 2003 RTD Embedded Technologies, Inc.

All rights reserved

Printed in U.S.A.

The RTD Logo is a registered trademark of RTD Embedded Technologies. cpuModule and utilityModule are trademarks of RTD Embedded Technologies. PS/2, PC/XT, PC/AT and IBM are trademarks of International Business Machines Inc. MS-DOS, Windows, Windows 98, Windows NT, Windows 2000 and Windows XP are trademarks of Microsoft Corp. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

Table of Contents

TABLE OF CONTENTS	4
INTRODUCTION	5
INSTALLATION	6
INSTALLATION OF THE DRIVER AND EXAMPLE PROGRAMS	6
THE DM6816 BOARD DRIVER.....	7
THE DRIVER CONFIGURATION.....	7
INSTALLING A SECOND BOARD IN THE SYSTEM	7
WORKING WITH OTHER BOARDS USING THE WINRT DRIVER	8
USING WINDOWS REGISTRY KEYS	9
THE DRIVER API FUNCTIONS	11
USING THE DRIVER WITHOUT THE WINDOWS-GUI	12
DRIVER API FUNCTION GROUPS	13
DRIVER INITIALIZATION FUNCTIONS.....	13
GENERAL BOARD CONTROL FUNCTIONS.....	13
DIGITAL I/O FUNCTIONS	13
INTERRUPT HANDLING	13
PWM CONTROL FUNCTIONS	13
USER TIMER-COUNTER FUNCTIONS	13
ERROR HANDLING FUNCTIONS.....	13
ALPHABETICAL DRIVER API FUNCTIONS REFERENCE	14
EXAMPLE PROGRAMS REFERENCE	23
WIN32 WINDOWS APPLICATIONS	23
LIMITED WARRANTY	24

Introduction

The DM6816 data Pulse Width Modulator Windows 98/NT4/2000/XP driver was designed for programmers who write Windows-based application programs with using the RTD's DM6816 board or DM6916 board. DM6816 and DM6916 are the same boards with different interface connector. Later we will refer both boards as DM6816.

The driver provides an Application Programming Interface with a lot of function calls to perform all the data acquisition tasks of the board users.

The board driver based on the BlueWater System's WinRT device driver kit.

There are example programs to demonstrate the various board features and the usage of the driver API. The example programs are written in Microsoft Visual C++ ver. 6.0.

Installation

Installation of the Driver and Example Programs

Before installing the driver and example program files, you need to install the DM6816 board in your PC. Please follow the instructions of the manufacturer, how to install the board in a computer.

To install the drivers for the DM6816 under Windows 98/2000/XP you need the installation diskette 0. The readme.txt file on this diskette describes the necessary installation steps.

For Windows NT4 the installation diskette 0 is not needed.

On the installation diskette 1 you can find the **Setup.exe** program, which installs on your PC the DM6816 board driver and example programs. The setup program automatically detects your operating system and installs the appropriate files on your PC.

After starting the setup, please follow the instructions on the screen to install the programs. You can select the directory where to install the files.

The setup also adds to the 'Start menu' under the 'Programs' folder of your Windows system the 'Real Time Devices USA' folder. It contains shortcuts to the example programs, the driver configuration utility and the readme.txt file.

The example programs will be installed in the target directory under the 'Examples' folder. This folder contains the example program sources for the DM6816 board, and the project files for Microsoft Visual C++ users to rebuild the programs. The example programs are compiled with the Microsoft Visual C++ ver. 6.0. In case of different version of Visual C is installed on your PC, please rebuild the executable files.

When you install the driver for the board, you have to set the IRQ and base address on the driver so they correspond to data set by jumpers P4 and P16 (IRQ) and DIP switch (base address). If you are using several boards each board needs to have its own unique IRQ and base address.

During the installation the Setup.exe program sets the default driver values:

Board 1: 0x300 base address, IRQ 0x11

If these settings are not proper to your PC configuration, you can modify the settings. On Windows NT use the **config.exe** program. On Windows 98/2000/XP you can do that using Windows Device Manager from Control Panel/System/[Hardware]/Device Manager.

Uninstallation: you can uninstall the DM6816 driver and example programs under the 'Control panel' with the 'Add/Remove Programs' tool.

The DM6816 board driver

The RTD's DM6816 board driver handles the hardware through the BlueWater's WinRT driver. The WinRT driver provides the low-level access to the board, and the RTD's `drv6816.dll` provides the device API for the programmers, and communicates with the hardware through the WinRT driver.

The RTD board drivers has multi-board feature. It means that it is possible to use more than one board in the system.

The Driver Configuration

As in the previous chapter mentioned, after installing the driver the default settings are used.

If these settings are not proper to your PC configuration, or there are other RTD boards used, on Windows NT you can modify the settings with the **Config.exe** program. This program is installed on your PC during the setup process and it can be started from the Start menu.

Program options:

- BOARD No. Choose the board (if there are more then one boards installed)
- IRQ. Choose the IRQ for the board
- BASE ADDRESS: Set the board's base address.
- SAVE: save configuration for this board in Windows registry
- EXIT: exit.

Under Windows 98/2000/XP same changes can be made using Control Panel/System/[Hardware]/Device Manager/WinRT Devices.

Installing a second board in the system

It is possible to use more than one board in the system using the RTD's drivers. Please follow the next steps to install a multi-board – system:

- Install the first board with all software and make sure that everything is working well.
- Place the second board in an empty slot in the PC, and switch on the computer.

Windows NT:

- After the system startup starts the installation program and answer 'YES' to the one more board installation question.
- If you need to uninstall one of several installed boards, you ALWAYS have to uninstall the highest number board.

Windows 98/2000/XP:

- Install another board using Disk0. Driver is ready to work with the additional board. Do not run the setup program on Disk1.

After installation of additional board, make sure that IRQ and Base address are set for both boards properly

Working with Other Boards Using the WinRT Driver

As you may have in your computer other RTD boards, which uses the WinRT driver, or a board from other manufacturer with WinRT, our installation program automatically detects the WinRT settings from the system registry file, and installs the appropriate (next) WinRT device. You can see an informational dialog box about this issue during the setup.

As an application written for the DM6816 board should know which WinRT device is assigned to the board, the installation program generates the file wrtdev.txt in the 'Examples' directory containing the actual device number. The example programs are reading this file at the startup to identify this number.

If the other board in your system, which uses the WinRT driver does not handle the situation when there is other WinRT-based application, install this board firstly, and then install the DM6816 which handles correctly this state.

Using Windows registry keys

The RTD board drivers are using the next registry keys:

Windows NT 4.0:

```
HKEY_LOCAL_MACHINE
|
|-- System
|   |
|   |-- CurrentControlSet
|       |
|       |-- Services
|           |
|           |-- WinRT
|               |
|               |-- WinRTdev0    (one per device)
|                   |
|                   |-- Parameters
|                       |
|                       |-- Section0
|
|               |-- WinRTdev1
|                   |
|                   |-- Parameters
|                       |
|                       |-- Section0
|
```

Windows 98 Enum key path:

```
HKEY_LOCAL_MACHINE
|
|-- Enum
```

Windows 98 Class key path:

```
HKEY_LOCAL_MACHINE
|
|-- System
|   |
|   |-- CurrentControlSet
|       |
|       |-- Services
|           |
|           |-- Class
|               |
|               |-- WinRTDevices
```

Windows 2000/XP device class information:

```
HKEY_LOCAL_MACHINE
|
|-- System
|   |
|   |-- CurrentControlSet
|       |
|       |-- Control
|           |
|           |-- Class
|               |
|               |-- {D695ED6A-630D-4D83-D8B-F1F0AC107AD0}
|               |
|               |-- 0001
```

Windows 2000/XP Plug and Play devices are enumerated under the Enum key:

```
HKEY_LOCAL_MACHINE
|
|-- System
|   |
|   |-- CurrentControlSet
|       |
|       |-- Enum
|           |
|           |-- PCI
```

The Driver API Functions

The resources on the DM6816 board can be accessed from Windows through the driver API (Application Programming Interface) functions. The executable code of these functions is located in the `drv6816.dll` file. To write applications using the API functions you must include the `reg6816.h` header file, and link the program with the `drv6816.lib` import library file. In the example programs you can find different examples how to use the driver.

Using the Driver without the Windows-GUI

This driver is based on the Win32 system, and can run only under Windows. However, the user, who is not familiar with the Windows Graphical User Interface, can use the driver API functions in Win32 console applications, which has an MS-DOS like text-mode interface.

In console applications is not necessary to use the Windows graphical environment, but all the driver's API functions are accessible.

The Microsoft Visual C++ compiler supports writing console applications.

Driver API Function Groups

Driver Initialization Functions

OpenBoard6816
CloseBoard6816

General Board Control Functions

InitBoard6816

Digital I/O Functions

DIOWrite6816

Interrupt Handling

ClearBoardIRQ6816
EnableBoardIRQ6816
EnableIRQSharing6816
InstallCallbackIRQHandler6816
InstallCounterIRQHandler6816
RemoveIRQHandler6816
GetIRQCounter6816
ReadBoardIRQStatus6816
SetIRQPolarity6816
LoadIRQRegister6816
IsP14IRQ6816
IsEPLD1IRQ6816
IsEPLD2IRQ6816
IsEPLD3IRQ6816

PWM control functions

DIOInitPWMChip6816
DIOLoadPWM6816
DIOEnablePWM6816
DIOEnableAllPWM6816
DIOSelectPWMClock6816

User Timer-Counter Functions

ClockDivisor6816
ClockMode6816
DoneTimer6816
ReadTimerCounter6816
SetUserClock6816

Error Handling Functions

GetErrorStatus6816

Alphabetical Driver API Functions Reference

C

ClearBoardIRQ6816

void ClearBoardIRQ6816 (RTDHANDLE hBoard)

Description:

This routine is used to clear the P14 IRQ circuitry.

Parameters:

hBoard:	device handle
---------	---------------

ClockDivisor6816

void ClockDivisor6816 (RTDHANDLE hBoard, int ATimer , unsigned int ADivisor)

Description:

The ClockDivisor procedure is used to set the divisor of a designated counter on the 8254 programmable interval timer (PIT). This procedure assumes that the counter has already been set to receive the least significant byte (LSB) of the divisor followed by the most significant byte (MSB).

Parameters:

hBoard:	device handle
ATimer:	0,1,2
ADivisor:	divisor

ClockMode6816

void ClockMode6816 (RTDHANDLE hBoard, int ATimer, int AMode)

Description:

The ClockMode procedure is used to set the mode of a designated counter on the 8254 programmable interval timer (PIT).

Parameters:

hBoard:	device handle
ATimer:	0,1,2
AMode:	0,1,2,3,4,5

CloseBoard6816

BOOL CloseBoard6816 (RTDHANDLE DeviceNumber, LPSTR szBuf)

Description:

This routine is used to close board driver.

Parameters:

DeviceNumber: device handle
szBuf: message buffer

Return value:

TRUE on success, FALSE on error

D

DIOEnableAllPWM6816

void DIOEnableAllPWM6816(RTDHANDLE hBoard, uchar8 Chip , uchar8 Enable)

Description:

This routine is used to enable or disable all 3 of the PWM outputs on the selected chip.

Parameters:

hBoard: device handle
Chip: 0 - 2
Enable: 0 = disabled
1 = enabled

DIOEnablePWM6816

void DIOEnablePWM6816(RTDHANDLE hBoard, uchar8 Chip , uchar8 SelectPWM,
uchar8 Enable)

Description:

This routine is used to enable or disable an individual PWM output on the selected chip.

Parameters:

hBoard: device handle
Chip: 0 - 2
SelectPWM: 0 – 2
Enable: 0 = disabled
1 = enabled

DIOInitPWMChip6816

void DIOInitPWMChip6816(RTDHANDLE hBoard,uchar8 Chip)

Description:

This routine is used to initialize the selected PWM chip.

Parameters:

hBoard:	device handle
Chip:	0 – 2

DIOLoadPWM6816

void DIOLoadPWM6816(RTDHANDLE hBoard, uchar8 Chip , uchar8 SelectPWM ,
uchar8 Data)

Description:

This routine is used to load the PWM duty cycle register. The duty cycle is Data/256.

Parameters:

hBoard:	device handle
Chip:	0 - 2
SelectPWM:	0 – 2
Data:	0 – 255

DIOSelectPWMClock6816

void DIOSelectPWMClock6816(RTDHANDLE hBoard, uchar8 Chip , uchar8 Clock)

Description:

This routine sets the PWM chip's clock source.

Parameters:

hBoard:	device handle
Chip:	0 - 2
Clock:	0 = 8 MHz 1 = Programmable clock

DIOWrite6816

void DIOWrite6816(RTDHANDLE hBoard, uchar8 Chip , uchar8 Data)

Description:

This routine writes the PWM chip's digital output bits.

Parameters:

hBoard:	device handle
Chip:	0 - 2
Data:	0 - 255

DoneTimer6816

void DoneTimer6816(RTDHANDLE hBoard)

Description:

Shut down timer operation initiated by ' SetUserClock6816'.

Parameters:

hBoard:	device handle
---------	---------------

E

EnableBoardIRQ6816

void EnableBoardIRQ6816 (RTDHANDLE hBoard, int AEnable)

Description:

This routine enables/disables the board P14 IRQ.

Parameters:

hBoard:	device handle
AEnable:	0 = disable
	1 = enable

EnableIRQSharing6816

void EnableIRQSharing6816 (RTDHANDLE hBoard, int AEnable)

Description:

This routine enables and disables the interrupt sharing feature of the board.

Parameters:

hBoard:	device handle
AEnable:	0 = Disabled
	1 = Enabled

G

GetErrorStatus6816

BOOL GetErrorStatus6816 (RTDHANDLE hBoard, LONG *ErrorCode, LPSTR ErrorString)

Description:

This routine is used to get the error code and string in case of errors.
The function clears the last error code.

Parameters:

hBoard:	device handle
ErrorCode:	Error Code
ErrorString:	Return error string

Return Value:

TRUE	if error occurred
FALSE	otherwise

GetIRQCounter6816

void GetIRQCounter6816(RTDHANDLE hBoard, IRQSetup *irqsetup)

Description:

This routine is used to get the counter incremented by IRQ.

Parameters:

hBoard:	device handle
irqsetup:	return parameters to caller

I

InitBoard6816

void InitBoard6816(RTDHANDLE hBoard)

Description:

This routine should always be called first.
This clears the board and variables the driver uses.

Parameters:

hBoard:	device handle
---------	---------------

InstallCallbackIRQHandler6816

BOOL InstallCallbackIRQHandler6816 (RTDHANDLE hBoard, isr_t handler)

Description:

This routine is used to install IRQ handler function.

Parameters:

hBoard:	device handle
isrT handler:	user callback function

Return value:

TRUE on success, FALSE on error

InstallCounterIRQHandler6816

BOOL InstallCounterIRQHandler6816(RTDHANDLE hBoard, IRQSetup *irqsetup)

This routine is used to install IRQ handler function.

Parameters:

hBoard:	device handle
irqsetup:	return parameters to caller

Return value:

TRUE on success, FALSE on error

IsEPLD1IRQ6816/ IsEPLD2IRQ6816/ IsEPLD3IRQ6816

uchar8 IsEPLD1IRQ6816(RTDHANDLE hBoard)

Description:

This routine checks the EPLD1-3 IRQ status bit.

Parameters:

hBoard:	device handle
---------	---------------

Return value:

1 if IRQ has been generated
0 if no IRQ

IsP14IRQ6816

uchar8 IsP14IRQ6816(RTDHANDLE hBoard)

Description:

This routine checks the P14 IRQ status bit.

Parameters:

hBoard: device handle

Return value:

1 if IRQ has been generated
0 if no IRQ

L

LoadIRQRegister6816

void LoadIRQRegister6816(RTDHANDLE hBoard,uchar8 Value)

Description:

This routine loads the interrupt register with one write operation.

Parameters:

hBoard: device handle
Value: value between 0-7

O

OpenBoard6816

BOOL OpenBoard6816(int num, LPSTR szBuf, BoardConfig *boardconfig)

Description:

This routine is used to open board driver.

Parameters:

num: board number to open
szBuf: Message buffer
boardconfig: Return board configuration

board configuration include:

int it_no; IRQ number used by board
int base_address; board base address
RTDHANDLE device_no; device handle used as argument for all other board control functions

Return value:

TRUE on success, FALSE on error

R

ReadBoardIRQStatus6816

int ReadBoardIRQStatus6816(RTDHANDLE hBoard)

Description:

This routine returns the IRQ status from the board.

Parameters:

hBoard: device handle

Returns:

the status register value

ReadTimerCounter6816

unsigned ReadTimerCounter6816 (RTDHANDLE hBoard, int ATimer)

Description:

Read the timer/counter count value.

Parameters:

hBoard: device handle
ATimer: number of the Timer in the range of 0..2.

Returns:

16-bit count value.

RemoveIRQHandler6816

void RemoveIRQHandler6816(RTDHANDLE hBoard)

Description:

This routine is used to uninstall IRQ handler function.

Parameters:

hBoard: device handle

S

SetIRQPolarity6816

void SetIRQPolarity6816 (RTDHANDLE hBoard, int Polarity)

Description:

This routine sets the P14 interrupt to use the positive edge or negative edge.

Parameters:

hBoard:	device handle
Polarity:	0 = positive edge
	1 = negative edge

SetUserClock6816

float SetUserClock6816 (RTDHANDLE hBoard, uchar8 Timer, float32 InputRate,
float32 OutputRate)

Description:

This routine sets the user timer counters.

Parameters:

hBoard:	device handle
Timer:	0 – 2
InputRate:	Input clock to selected timer.
OutputRate:	Desired output rate from selected timer.

Returns:

The actual programmed clock frequency.

Example Programs Reference

Win32 Windows applications

Name	Feature	Remarks
Digital	Sample program that demonstrates how to program digital I/O.	This program toggles the digital output lines timed by the internal timer of Windows.
Interrupt	Sample program that demonstrates how to use the timer as an interrupt source and how to write an Interrupt Service Routine (ISR).	This program sets up the User Timer to generate interrupts at a selected rate. An interrupt service routine (ISR) is provided that increments the PWM0.0 (Pin 47 on P2) on every interrupt. The jumpers on P5 to set up the clock should be set to the factory settings as shown in the manual.
Pwm	Sample program that demonstrates how to program the PWM outputs.	All Pulse Width Modulators will output a Square wave (50% duty cycle). The user can Start or Stop the operation with pressing a button.
Timers	Sample program that demonstrates how to program the user timer.	Sample program demonstrating how to program the 8254 programmable Interval timers (PITs). Counters 0 and 1 are programmed to output a 200 Hz pulse train. Counter 2 is used to count these pulses. The result is converted into seconds and printed on the screen. The jumpers on P5 should be set at factory default as shown in the manual.

Limited Warranty

RTD Embedded Technologies, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from RTD Embedded Technologies, INC. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, RTD Embedded Technologies will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to RTD Embedded Technologies. All replaced parts and products become the property of RTD Embedded Technologies. Before returning any product for repair, customers are required to contact the factory for an RMA number.

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, "acts of God" or other contingencies beyond the control of RTD Embedded Technologies), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN RTD Embedded Technologies. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND RTD Embedded Technologies EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER'S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL RTD Embedded Technologies BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College PA 16803-0906
USA
Our website: www.rtd.com